IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 18, 2019, accepted January 4, 2020, date of publication January 27, 2020, date of current version February 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2968627

Using Smartphones to Enable Low-Cost Secure Consumer IoT Devices

ROSS MCPHERSON AND JAMES IRVINE, (Senior Member, IEEE)
Department of Electronic and Electrical Engineering, Royal College Building, Glasgow G1 1XW, U.K.

Corresponding author: Ross McPherson (ross.mcpherson@strath.ac.uk)

This work was supported by the Aggregators as diGital Intermediaries in Local Electricity markets under Project EP/S003088/1: EPSRC/ESC.

ABSTRACT This paper proposes a solution for low-cost consumer IoT devices to employ end-to-end security without requiring additional hardware. Manufacturers of consumer IoT devices often sacrifice security in favour of features, user-friendliness, time to market or cost, in order to stay ahead of their competitors. However, this is unwise, as demonstrated by recent hacks on consumer IoT devices. Low-cost embedded devices struggle to create suitable entropy for key generation; on the other hand, smartphones are both abundant and have multiple sources of entropy for strong key generation. The proposed architecture takes advantage of these properties and offloads key generation and transfer to the user's smartphone, removing the need for constrained IoT devices to perform public key infrastructure and generate symmetric keys. The authors implemented the design on a $1 general-purpose microcontroller and then analysed the performance. The design allows all communication to and from the device to be encrypted while being simple to setup, low-cost and responsive without any additional manufacturing cost. The architecture presents a general solution, which could be implemented on any microcontroller. Since the architecture does not require any additional hardware, it can be retroactively applied to deployed devices through a firmware update.

INDEX TERMS Internet of Things, security, encryption, wireless communication, microcontrollers.

I. INTRODUCTION

Recent developments in consumer behaviour have driven an increase in the market for Home Automation devices. These devices promise to make our lives more convenient and automated. For example, lights can be turned on and off using a voice assistant, a smartphone, or through automatic triggers such as when approaching the home. Although automated systems have been available for many years, as technology has improved, and the cost of devices and difficulty of installation has decreased, consumer Internet of Things (IoT) devices have seen a rapid increase in demand, particularly in a domestic setting [1]. However, this convenience and low-cost comes with a risk. There have been numerous recent instances of domestic internet-connected devices being compromised [2]–[5]. However, as the domestic consumer market is price sensitive, this introduces a challenge to make the devices secure. Implementing security features could lead to higher costs for manufactures [6]. This paper will focus on domestic low-cost Wi-Fi IoT devices and explore approaches which can be taken to introduce robust security.

The associate editor coordinating the review of this manuscript and approving it for publication was Ana Lucila Sandoval Orozco.

II. BACKGROUND

A. MARKET CONSTRAINTS

As the market for Internet of Things (IoT) devices has been steady growing, a recent UK Parliamentary Office of Science and Technology report [7], listed economic drivers as one of the main reasons for poor cyber security for consumer devices. The EU Agency for Cybersecurity, academics, and industry have all expressed concerns that manufacturers prioritise cost, user experience and time to market over security [8]–[12]. Moreover, a 2017 survey found that 42% manufacturers' consumers were not willing to pay for security, and the next largest section at 28% only willing to pay a 1-10% premium [13]. This means that security must be provided at no additional hardware cost.

B. SECURITY

Advanced Encryption Standard (AES), uses one shared key to quickly and efficiently encrypt and decrypt data, even on low-cost devices [14], [15]. This can be achieved using either a purely software or hardware accelerated version. The main shortfall for this type of cryptography is the requirement to have the same key on both end devices, leading to the challenge of sharing the key securely between the

devices [16]. The conventional way to overcome this problem is by employing hybrid encryption. Public key encryption is used to set up a secure channel, which is used to transfer the shared symmetric key [17], [18]. This approach works well in high-performance computing devices such as smartphones but does not scale down well to low-cost embedded hardware.

C. SECURITY NECESSITY

Consumer IoT devices can be separated into two categories: devices which can natively support security and devices which require additional hardware to communicate securely. The former is made up from comparatively high performance devices, commonly featuring ARM chips and/or the ability run full operating systems. One such device is the Google Chromecast, which features an AES hardware accelerator and a random number generator [19], and so is capable of implementing public key infrastructure and generating a suitably random systematic key to encrypt future communications.

In the second category, there are much more constrained devices which are only designed for a single limited purpose, which achieve their primary functionality using a much lower performance computing hardware. These lower performance chips commonly do not feature hardware AES accelerators or hardware random number generators. Moreover, since these devices follow a predictable series of events, they struggle to generate sufficient entropy to create secure keys. One example of a constrained device is the Belkin Wemo Switch, which offered customers the ability to remotely turn on and off any device plugged into a mains socket. Users could also use their smartphone to check if they had switched off a particular appliance, trigger their lights to turn on when arriving home, or programmatically schedule devices to turn on or off — saving the consumer money and time [20]. Nonetheless, shortly after the product was released, various exploits were discovered in the device's architecture resulting in Belkin's customers being advised not to use the product anymore [21]. These exploits were possible because Belkin had employed password-based encryption to generate the AES key, where the parameters of key were predictable [22].

D. ENTROPY GENERATION

The security of a communication channel depends on the unpredictability of the keys used to encrypt that channel; therefore, suitability random entropy sources are required to ensure that strong keys are generated. If an attacker could predict the credentials, even the most sophisticated security scheme would be compromised by simply guessing the key. With traditional communication, this is not an issue as modern computers and smartphones already get entropy from a variety of diverse sources, which can be combined [23]–[26]. On the other hand, generating suitability robust keys in low-cost IoT devices is more challenging. These devices commonly run a limited set of instructions and have minimal user interaction [27]. In response, the research community has investigated various potential solutions to enable constrained embedded devices to generate suitable entropy.

The LoRaWAN alliance attempted to solve this problem of single-purpose devices not being suitably complex to generate entropy by reusing existing hardware. Noting that these devices would have to have an antenna to send and receive messages, the antenna was employed as a sensor, recording the received signal strength [28]. This was assumed to be suitably random, but researchers have discovered that by jamming or otherwise influencing the signal strength, it could become more predictable [29]. This demonstrated that although existing sensors may be available, and could provide semi-random data, if the sensor could be influenced, this would make for a poor entropy source.

Academic researchers did discover several ways for embedded devices to generate sufficient entropy when employing additional hardware. Voris *et al.* [30] investigated using various individual sensors, including an accelerometer, magnetometers, proximity sensor, photometers, thermometer, microphone and discovered that each of these would create some entropy. However, all were able to be influenced by external interference, with the exception of the accelerometer.

The researchers went onto discover that any attempts to reduce the entropy generated by the accelerometer resulted in the entropy increasing. This would suggest that using an accelerometer would be a solution to providing entropy in high constrained devices. Similarly, crypto authentication coprocessors which feature AES hardware accelerators, and hardware random number generator, would be able to create sufficient entropy. Finally, researchers discovered that certain specific implementations of microcontrollers could take advantage of their hardware to generate entropy. However, it is important to note all of these solutions work for particular cases, but are not a general solution. As devices which did not already feature these properties, there would be an increase in the raw material cost of the device.

E. LEGISLATION

Legislation is catching up with the problem. The recent California Senate Bill No. 327 [31] states that manufacturers must make a reasonable attempt at cybersecurity before selling a device in California. While this only directly affects California, manufacturers may apply these changes universally to avoid maintaining two different versions of a device. On the other hand, the United Kingdom's recent Code of Practice for Consumer IoT Security [32], outlines a number of improvements for IoT security, but is currently only voluntary. Similarly, the European Telecommunications Standards Institute (ETSI) has also outlined 13 practices that manufacturers can optionally follow [33]. Regulation 2019/881 also created the European Union Agency for Cybersecurity, where there remit is to create certification frameworks and governance for the EU on cyber security matters [34]. However as of the time of writing, there have been no defined rules. In the United States the Internet of Things Cybersecurity Improvement Act of 2019 [35], aims to use the purchasing power of the US federal government to persuade IoT manufactures to improve their device security. However, this bill is currently only

making its way through the legislative process, and would also only be voluntary.

III. ARCHITECTURE

Our solution uses a smartphone to generate keying material securely, which it then passes to the consumer IoT device, and also to the IoT device's server. Smartphones are ubiquitous with 87% of the UK population [36], and 95% of the target demographic for Smart Homes owning at least one smartphone [37]. They have powerful processors, multiple sensors, rich user interaction and run multiple processes and so are very capable of generating sufficient random data to provide suitably random keys [26]. Transferring the key to the IoT device also takes advantage of existing technology, given that by their nature IoT devices already have an internet connection.

Google employs a superficially similar setup procedure with their popular Wi-Fi-based Chromecast devices, where a smartphone is used in the setup procedure to connect the IoT device to the domestic Wi-Fi network. This approach has also been patented for household appliances [38]. However, Chromecast initiates a TLS connection as it features a Marvell DE3005-A1 system-on-chip [19], which contains a hardware random number generator and AES hardware accelerator. Low-cost, low-powered devices do not feature any of this additional hardware.

A. SETUP PROCEDURE

The secure initialisation protocol operates as follows:

1) The device is factory provisioned with an (ideally unique) wifi credential, made available to the user in the packaging
2) On initial activation, the device sets up a private network using this credential, and the user connects to it using their smartphone
3) The smartphone seeds random data and generates an AES key
4) The smartphone shares this key with the IoT device, along with the credentials for the network the IoT will use to connect to the Internet
5) The smartphone and IoT device disconnects from the private network and reconnects to the Internet
6) The smartphone and IoT device establish a secure connection between each other over the local network using symmetric encryption with the key the smartphone generated and shared
7) The smartphone shares the IoT device's key securely with the remote server.

On start-up, the device will broadcast its Wi-Fi access point, using a default SSID and device-specific password, inline with the UK's Code of Practice for Consumer IoT Security [32]. The passwords will be provisioned into the devices at the factory and will be optionally supplied inside the product box in the form of a QR code. An additional advantage of this approach would be the automatized usage of 63 character long Wi-Fi passwords while removing the potential for human error during input.

Interception of the package en route to the user is possible, where a malicious actor opens the package and records the supplied QR code to intercept the transmission of the AES key during device setup, and therefore compromises the device. However, several additional steps can be taken to mitigate these risks.

1) The Wi-Fi transmit power of the IoT device can be turned down to its lowest power, 0.25 dBm in the case of the ESP8266, so that the signal disappears into the noise floor when attempting to detect at a distance greater than 20 cm away. The Wi-Fi adaptive power control of Android phones will cause the corresponding smartphone transmit power to decrease accordingly
2) Configuring the Wi-Fi module to support a single connection; therefore, only a malicious attacker or a legitimate user can be on the temporary Wi-Fi at any one time.
3) sealing the QR code in a tamper-evident envelope

Additionally, the 2.4 GHz Wi-Fi band is highly congested [39], and, therefore, it would be challenging to detect the correct signal through a malicious attacker waiting outside the property. Furthermore, the temporary Wi-Fi network would only be configured to broadcast for approximately one minute, which would be less than the typical time for a button press Wi-Fi Protected Setup (WPS).

An accompanying app downloaded from an application marketplace, e.g. the Google Play Store, which would generate an AES key, and automate the connection to the private network. Initiating this process, the user would scan the QR code, the app would extract the credentials and switch the smartphone's Wi-Fi from the home network to the IoT device's generated network. The smartphone would then generate an AES key and transfer it to the IoT device. The user would then be prompted for any specific information, e.g. a personalised device identifier and the domestic Wi-Fi credentials for the IoT device to join the network.

After the key is shared, both devices (re)join the domestic network. Once the IoT device and the smartphone can see each other on the network, the joining process will be considered successful. Finally, the smartphone will take the user through the optional stage, linking the device to an external server, for out of the house control. The smartphone will transfer the shared key by setting up an SSL/TLS connection to the server to securely transfer the AES key. The phone, the switch and server should now be able to contact each other and encrypt/decrypt all messages. As illustrated in Figure 1, the proposed architecture is based on a local interaction, with keys then being sent to a remote location, rather than trying to provision a unique key from a remote location. This would provide the following advantages:

- If a user does not wish to control their device from outside their home, no external communication is required.
- If the manufacturer's servers were down or decommissioned, the product would continue to operate,

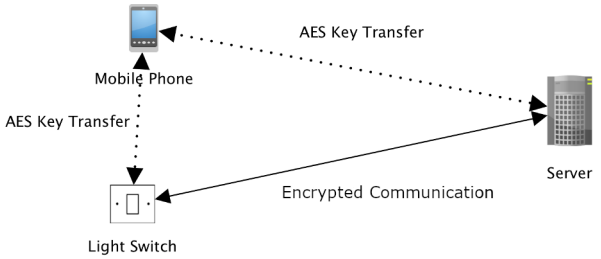

FIGURE 1. Setup Architecture.

whereas similar devices have previously lost functionality in these situations. In all these instances the device could default to running locally and still be locally functional.

- There is no requirement on the product manufacturer to generate and manage the distribution of keys.
- By transferring the key over a short term private network, a compromised device on the domestic network could not overhear the key. Transferring the key after the device had joined the domestic network would allow an interception.
- The entry of high entropy keys into the IoT device is greatly simplified for the user

B. INTERNAL USER COMMUNICATION

The mobile application achieves internal communication without a centralised hub by monitoring the network. On detection of a new switch, the mobile application alerts the user to its presence, prompting them to start the installation process. If any switches have lost their previously assigned address, periodic searches would also find and record any change in the IP address. The downside to this approach is that any malicious device on the network could detect the IoT device's presence through its fingerprint. However, considering the device could be detected through several other techniques [40]–[42], it was reasoned that this would make a minimal difference, considering all communication would be encrypted.

C. EXTERNAL USER COMMUNICATION

Communication with the IoT device when on separate networks is managed via an API endpoint. An authorised trigger, the user's phone, a voice assistant, such as Google Home, Alexa, Siri, or automation apps, such as IFTTT, tasker, zapier, can all connect to the manufacturer's server to post an HTTP request to the relevant endpoint, and update the state of the device. The IoT device will then periodically poll the central server and update accordingly.

IV. IMPLEMENTATION

This purposed architecture has been implemented, creating an IoT smart light switch, constructed using an MSP430G2553 microcontroller and an ESP8266 Wi-Fi chip. The Smart Switch can be controlled both from physical buttons on the light switch and through an Android application. The app works locally on the local network, and remotely via a centralised server. Linking with a centralised server also allows the switch to be part of a more extensive system. For example, an IFTTT recipe was also set up in conjunction with a Google Home assistant, so that the light could be controlled via voice.

A. DEVICE SETUP

As described in section III-A the setup of the Smart Switch is done by wiring the switch in and then joining the private Wi-Fi network it creates to transfer the AES key generated on the Android phone. The app can also to be used to personalise details such as a name for the light, so the user knows which light they are modifying in the app, e.g. master bedroom. Once this has completed, the user can decide if they want to authorise external connections. If they agree, the app makes a secure connection to the central server to transfer the details of the light and associated AES key to the user's remote account.

B. INTERNAL COMMUNICATION

Given that the system was designed to operate without an external server or local hub, the Android phone uses mDNS to detect devices advertising themselves as Smart Switches and will encrypt the message using the associated AES key. The receiving switch confirms the message authenticity by checking the HMAC, before decoding and acting on the payload.

C. EXTERNAL COMMUNICATION

Any time there is a brightness change, either by the app or physical change in the light switch, the device sends an HTTP Post request to the server updating the status of the light held remotely. Once the user opens the mobile application, an HTTP request is issued to adjust the display to reflect the exact brightness. On a change of state request, a request is made to the server updating the state. The smart switch device will poll the server every 30 seconds, and if a dispute is detected, the physical buttons override the server status.

D. ENCRYPTION

Predictable messages, such as on and off, could infer occupancy of the home by analysing the pattern. Moreover, if these messages remained the same each time, it would be possible to record the messages for a replay attack. Therefore, it was decided to implement AES counter mode on the server, Android app & Smart Switch, obscuring the messages, and by implementing counters similar to those stipulated in the LoRaWAN specification preventing replay attacks [28]. To protect against bit-flipping attacks a message authentication code (MAC) was also calculated, by each device and included in the sent message. The receiving device checks the MAC before decryption the payload.

TABLE 1. Software encryption operation.

	50000 operations		1 operation	
Clock Speed (MHz)	Time (seconds)	Power (uJ)	Time (ms)	Power (uJ)
1	632825	1090476.2	12.656	21.809
8	96072	882227.7	1.921	17.644
12	62287	844482.4	1.245	16.890
16	47569	831062.5	0.951	16.621

TABLE 2. Software decryption operation.

	50000 operations		1 operation	
Clock Speed (MHz)	Time (seconds)	Power (uJ)	Time (ms)	Power (uJ)
1	632,850	1,916,670.00	23.33	38.33
8	148,350	1,337,175	2.97	26.74
12	96,530	1,289,929	1.93	25.80
16	47569	831062	1.48	25.53

TABLE 3. Power consumption at different clock speeds.

Clock Speed (MHz)	Time (seconds)	Power (uJ)
1	10	17347
8	10	91875
12	10	135698
16	10	175135
LPM3	10	4826.5

V. RESULTS

Although the smart switch implementation was made as a proof of concept, the proposed architecture would suit any low-cost IoT device. With this in mind the time and energy requirements of the system were measured for running an AES128 software implementation on an MSP430G2553. During testing, it was discovered that the measurement equipment was not accurate enough to measure a single encryption/decryption operation. Therefore, it was decided to execute the process 50,000 times and extrapolate for what would be the requirements for a single operation of each encryption and decryption. It was also decided to vary the clock rate of the MSP430G2553 chip to compare the time and energy consumed. This data is shown for encryption in table 1 which took 15,725 cycles to complete a single AES encryption operation.

The software implementation for decryption was also profiled. Although it was found to require 23,470 cycles - an increase over the encryption algorithm. The testing showed that both AES128 encryption and decryption were viable for greater than 25 ms response times. This is notably less than the average of 250ms human response time [43]. The timings and energy consumption are detailed in table 2. The authors note that although the power consumption was lower at lower clock frequencies, the increased time requirements of the 1 MHz clock operation results in using the highest total consumption. Therefore, depending on the application, it may be more beneficial to operate at the highest power consumption — 16MHz —, and then enable one of the built-in low-power modes. On the other hand, if the system is designed to be continually running without any low-power modes enabled, it would save power by running at 1MHz. A table detailing the power consumption at different clock speeds at a fixed interval of 10 seconds, along with the consumption of low-power mode 3 in table 3.

As outlined above, three of the issues securing low-cost IoT devices are; key generation, setting up public-key encryption and remaining responsive. Offloading the generation to a smartphone solves the first issue. Using the same smartphone to transfer the key solves the second issue. Finally, by demonstrating that encryption and decryption are achievable on an MSP430G2553 in a reasonable time, the proposed architecture solves all of the outlined problems, enabling a $1 microcontroller to use AES128.

VI. FUTURE WORK

The proposed architecture works well for a single user or multiple users sharing the same set of keys. However, if each user required dedicated keys, for example to provide zonal control over the lights. Each user would have to repeat the setup process for each light, they are allowed to control, a time-intensive task. The simple solution to this would be to have each new user detect all the switches they can see on the network through mDNS, and generate a new key for each one. However, since these keys would then be transferred over the home Wi-Fi network, if this were compromised, then each of these keys would also be compromised. A more sophisticated solution would involve creating a master user. The master user would create master AES keys, who would initially setup up all the devices. Once control is to be delegated to an new individual sub-keys would be generated and transferred to each of the required switches encrypted with the master key. The master user would then separately send each sub-key to the user's phone using Bluetooth or NFC. However, storing further keys in these constrained would require more resources, and would be more complex for users, therefore it would be the vendors decision how to proceed in their specific case.

The proposed architecture was tailored to Wi-Fi devices because the majority of the domestic IoT device market use Wi-Fi. Despite this, a similar idea could be applied to wearable devices, which predominantly use Bluetooth to connect to a smartphone. Furthermore, by modifying the design to relay data via a local intermediary, such as a hub, additional communication interfaces not commonly found on smartphones could also be used. For example, Philips Hue uses Wi-Fi to communicate with a hub, and then Zigbee to the bulb. The hub acting as a relay for the phone generated key would allow each Phillips Hue lightbulb to end-to-end encrypt its traffic. Further work could be undertaken to implement this modification.

VII. CONCLUSION

The created architecture allows low-cost IoT devices to have suitably random keys without additional dedicated hardware. By offloading key generation and transfer to the user's

smartphone, there is no requirement for additional components to generate entropy. Furthermore, by offloading key transfer, powerful processors, which would mostly sit idle, are not required for secure public-key encryption. The paper also detailed the power and timing requirements of such a system. We have shown that with our scheme there would be less than 25 ms delay in responsiveness due to the encryption overhead. The architecture also provides a user-friendly way of setting up the device securely, and combined with recent legislation on IoT device security, demonstrates this architecture would be beneficial. Specifically, the architecture allows for end-to-end encryption in low-cost IoT devices, protecting from 'man in the middle' attacks, UPnP vulnerabilities, and malicious devices on the same network. Applying this architecture to the Belkin Wemo Switch would have prevented these devices being exploited in the way detailed above. Moreover, as the architecture requires no additional hardware, it could be deployed to existing devices through a firmware update. Finally, as the architecture is platform-agnostic, it could be deployed for any backend web server, and with a small modification to other forms of IoT devices without Wi-Fi connectivity.

REFERENCES

[1] S. Singh, "Consumer IoT market worth $104.4 billion by 2023," Markets and Markets, Northbrook, IL, USA, Tech. Rep. SE 6810, Dec. 2018.
[2] R. Nigam. (Jan. 22, 2016). *CVE-2015-4400: Backdoorbot Network Configuration Leak on a Connected Doorbell*. FortiNet. [Online]. Available: https://www.fortinet.com/blog/threat-research/cve-2015-4400-backdoorbot-network-configuration-leak-on-a-connected-doorbell.html
[3] D. Bryan. (Jan. 8, 2013). *Lack of Web and API Authentication in Insteon Hub*. Trustwave SpiderLabs. [Online]. Available: https://www.trustwave.com/en-us/resources/security-resources/security-advisories/?fid=18876
[4] S. Margaritelli. (Oct. 9, 2016). *Reversing the Smarter Coffee IoT Machine Protocol to Make Coffee Using the Terminal*. [Online]. Available: https://www.evilsocket.net/2016/10/09/IoCOFFEE-Reversing-the-Smarter-Coffee-IoT-machine-protocol-to-make-coffee-using-terminal/
[5] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the IoT zombie armies," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2017, pp. 267–272.
[6] T. August, D. Dao, and K. Kim, "Market segmentation and software security: Pricing patching rights," *Manage. Sci.*, vol. 65, no. 10, pp. 4575–4597, Oct. 2019.
[7] (Feb. 2019). *Cyber Security of Consumer Devices*. [Online]. Available: https://researchbriefings.parliament.uk/ResearchBriefing/Summary/POST-PN-0593
[8] D. Barnard-Wills, L. Marinos, and S. Portesi, "Threat landscape and good practice guide for smart home and converged media," Eur. Union Agency Netw. Inf. Secur. (ENISA), Attiki, Greece, Tech. Rep. 978-92-9204-096-3, 2014.
[9] C. Lévy-Bencheton, E. Darra, G. Tétu, G. Dufay, and M. Alattar, "Security and resilience of smart home environments good practices and recommendations," Eur. Union Agency Netw. Inf. Secur. (ENISA), Attiki, Greece, Tech. Rep. 978Ű92-9204-141-0, 2015.
[10] P. Morgner and Z. Benenson, "Exploring security economics in IoT standardization efforts," 2018, *arXiv:1810.12035*. [Online]. Available: https://arxiv.org/abs/1810.12035
[11] Martyn Thomas CBE FREng. (Feb. 2017). *Written Evidence to the Joint Committee on the National Security Strategy*. UK National Security in a Digital World. [Online]. Available: http://data.parliament.uk/writtenevidence/committeeevidence.svc/evidencedocument/national-security-strategy-committee/cyber-security-uk-national-security-in-a-digital-world/written/47405.pdf
[12] *Common Position on Cybersecurity*, Infineon, Neubiberg, Germany, Dec. 2016.
[13] H. Bauer, O. Burkacky, and C. Knochenhauer. (May 2017). *Security in the Internet of Things*. McKinsey & Company. [Online]. Available: https://www.mckinsey.com/~/media/McKinsey/Industries/Semiconductors/Our%20Insights/Security%20in%20the%20Internet%20of%20Things/Security-in-the-Internet-of-Things.ashx
[14] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, "Report on the development of the advanced encryption standard (AES)," *J. Res. Nat. Inst. Standards Technol.*, vol. 106, no. 3, pp. 511–577, 2001.
[15] J. Daor, J. Daemen, and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*. Springer-Verlag, 2002.
[16] R. C. Merkle, "Secure communications over insecure channels," *Commun. ACM*, vol. 21, no. 4, pp. 294–299, Apr. 1978, doi: 10.1145/359460.359473.
[17] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.
[18] T. Dierks, *The Transport Layer Security (TLS) Protocol Version 1.2*, document RFC 5246, Internet Requests for Comments, RFC Editor, Aug. 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5246.txt
[19] *88DE3006 High-Definition Secure Media Processor System-on-Chip (SoC)*, Marvell, Hamilton, Bermuda, 2012.
[20] Belkin. (Feb. 4, 2019). *Wemo Insight Smart Plug*. [Online]. Available: https://www.belkin.com/us/p/P-F7C029/
[21] CERT Coordination Center. (2014). *Belkin Wemo Home Automation Devices Contain Multiple Vulnerabilities*. [Online]. Available: https://www.kb.cert.org/vuls/id/656302/
[22] H. Liu, T. Spink, and P. Patras, "Uncovering security vulnerabilities in the Belkin WeMo home automation ecosystem," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2019, pp. 894–899.
[23] Z. Gutterman, B. Pinkas, and T. Reinman, "Analysis of the Linux random number generator," in *Proc. IEEE Symp. Secur. Privacy*, May 2006, pp. 371–385.
[24] N. Sullivan. *Ensuring Randomness With Linux's Random Number Generator*. Cloudfare. Accessed: Jan. 15, 2019. [Online]. Available: https://blog.cloudflare.com/ensuring-randomness-with-linuxs-random-number-generator/
[25] A. Vassilev and R. Staples, "Entropy-as-a-service: Unlocking the full potential of cryptography," *Computer*, vol. 49, no. 9, pp. 98–102, 2016.
[26] J. Loutfi, A. Chehab, I. H. Elhajj, and A. Kayssi, "Smartphone sensors as random bit generators," in *Proc. IEEE/ACS 11th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2014, pp. 773–780.
[27] K. Munro. (Dec. 7, 2016). *IoT Encryption: The Challenge of Missing Entropy*. [Online]. Available: https://www.pentestpartners.com/security-blog/iot-encryption-the-challenge-of-missing-entropy/
[28] N. Sornin and A. Yegin, "Lorawan specification v1.1," LoRa Alliance, Beaverton, OR, USA, Tech. Rep. 1.1, 2017.
[29] S. Tomasin, S. Zulian, and L. Vangelista, "Security analysis of LoRaWAN join procedure for Internet of Things networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Mar. 2017, pp. 1–6.
[30] J. Voris, N. Saxena, and T. Halevi, "Accelerometers and randomness: Perfect together," in *Proc. 4th ACM Conf. Wireless Netw. Secur. (WiSec)*. New York, NY, USA: ACM, 2011, pp. 115–126, doi: 10.1145/1998412.1998433.
[31] California Senate Legislature. (2018). *Senate Bill No. 327*. ch. 886. [Online]. Available: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB327
[32] (2018). *Code of Practice for Consumer IoT Security*. [Online]. Available: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/773867/Code_of_Practice_for_Consumer_IoT_Security_October_2018.pdf
[33] "Cyber security for consumer Internet of Things," Eur. Telecommun. Standards Inst., Cedex, France, Tech. Rep. 1.1.1, Feb. 2019.
[34] The European Parliament and the council of the European Union. (Jun. 2019). *Regulation (EU) 2019/881 of the European Parliament*. Brussels, Belgium. no. 881. [Online]. Available: http://data.europa.eu/eli/reg/2019/881/oj
[35] M. Warner, "Internet of Things cybersecurity improvement act of 2019," 116th Congr. (2019), Senate United States, Washington, DC, USA, Tech. Rep. 734, 2019. Accessed: Nov. 5, 2019. [Online]. Available: https://www.congress.gov/bill/116th-congress/senate-bill/734/

[36] Deloitte. (2018). *Global Mobile Consumer Survey 2018*. Accessed: Mar. 19, 2019. [Online]. Available: http://www.deloitte.co.uk/mobileuk/assets/img/download/Global-Mobile-Consumer-Survey-2018-UK-Cut-State-of-the-smartphone.pdf
[37] C. W. Tom Hargreaves. (2013). *Who Uses Smart Home Technologies? Representations of Users by the Smart Home Industry*. European Council for an Energy Efficient Economy. p. 1775. [Online]. Available: https://www.eceee.org/library/conference_proceedings/eceee_Summer_Studies/2013/6-appliances-product-policy-and-ict/who-usessmart-home-technologies-representations-of-users-by-the-smart-homeindustry/2013/6-241-13_Hargreaves.pdf/
[38] M. P. Ebrom, "Method of connecting an appliance to a WiFi network," EP Patent 2 814 273 A1, Jun. 10, 2013.
[39] Z. Hays, G. Richter, S. Berger, C. Baylis, and R. J. Marks, "Alleviating airport WiFi congestion: An comparison of 2.4 GHz and 5 GHz WiFi usage and capabilities," in *Proc. Texas Symp. Wireless Microw. Circuits Syst.*, Apr. 2014, pp. 1–4.
[40] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "DEFT: A distributed IoT fingerprinting technique," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 940–952, Feb. 2019.
[41] D. Nouichi, M. Abdelsalam, Q. Nasir, and S. Abbas, "IoT devices security using RF fingerprinting," in *Proc. Adv. Sci. Eng. Technol. Int. Conf. (ASET)*, Mar. 2019, pp. 1–7.
[42] H. Jafari, O. Omotere, D. Adesina, H.-H. Wu, and L. Qian, "IoT devices fingerprinting using deep learning," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2018, pp. 1–9.
[43] A. Jain, R. Bansal, A. Kumar, and K. Singh, "A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students," *Int. J. App. Basic Med. Res.*, vol. 5, no. 2, p. 124, 2015.

ROSS MCPHERSON received the B.Eng. degree (Hons.) in computer and electronic systems from the University of Strathclyde, Glasgow, U.K., in 2016, where he is currently pursuing the Ph.D. degree in cyber security in the internet of things devices. He is focusing on the practical development and usage of devices without detracting from the existing user experience. He has previously worked in the financial, utility, and mobile communication industries. During his studying time, he has created private mobile and low-power wide area communication networks, and retrofitted existing hardware to enable secure remote communication.

JAMES IRVINE (Senior Member, IEEE) received the Ph.D. degree from the University of Strathclyde, Glasgow, in 1994. He currently leads the communications and system integration theme of the Power Networks Demonstration Centre, part of the EEE Department at the University of Strathclyde. He is a co-author of two books, over 140 journal and conference papers, and seven patents. He has testified in the U.K. and the Netherlands on multiple cases involving cellular radio technology. His research focusses on mobile communication and security, in particular on resource allocation and coding theory.

Dr. Irvine is a member of the IEEE Blockchain Initiative Steering Group, chairing the events activity. He is an elected Board member of the IEEE Vehicular Technology Society. He was active for many years in the U.K. Mobile VCE research programme, as an Academic Coordinator of the Instant Knowledge work area on privacy and trust in new personal communication services, and he led security work in the previous two MVCE Core programmes. His conference activities include the General Chair of VTC2015-Spring, the TPC Chair of VTC2004-Spring, and the TPC Co-Chair of ISWCS2007. He was the President of the IEEE Vehicular Technology Society, from 2008 to 2009, and the founding Editor-in-Chief of the *IEEE Vehicular Technology Magazine*.